📋 API CA Prep Day 3: CRUD Routes & Error Handling Middleware

Table of Contents

1. Core Requirements

Today you'll refactor your Express application to handle errors centrally with middleware. You'll implement CRUD routes for Events, properly protecting endpoints with JWT authentication, and ensure your Jest integration tests are properly configured.

2. Middleware Refactoring

Clean Up

- Delete the `views` folder.
- Remove existing error handling middleware in `app.js`.

Error Handling Middleware

Create centralized error handling middleware in `middleware/errorHandler.js`:

- Handle Sequelize validation and foreign key errors.
- Handle generic errors and HTTP errors using the existing `http-errors` package.

TIP: Use this repo to help guide you.

Register Middleware

- In `app.js`, register your new error handling middleware after your routes.

Note: Ensure you also register this middleware in your Jest mock app setup used for Supertest.

3. CRUD Routes

Create a new route file: `/routes/events.js`.

Routes to Implement:

- **GET /events** (Public)
    - Retrieves all events.

- **GET /events/mine** (Protected)
    - Retrieves events created by the logged-in user.
- **POST /events** (Protected)
    - Creates a new event associated with the logged-in user.
- **PUT /events/:id** (Protected)
    - Updates an event by ID if owned by the logged-in user.
    - If the ID of the logged in user is different to the `userId` of the event, return a 403 Forbidden.

Route Requirements:

- Properly use the middleware pattern (use `next(error)` or `next(createError(code, message))` to forward errors to your centralized error middleware).
- Clearly handle expected HTTP errors (`400`, `401`, `404`).
- Write integration tests to verify this behaviour. This repo can help remind you of the paths we test for with routes.

4. JWT Middleware Update

Update your existing JWT authentication middleware to:

- Set `req.user = decoded.sub` after token validation. This is so you can use it in the routes that need the logged in user's ID.

Ensure this middleware is correctly applied to protected routes in your events routes file.

5. Test App Setup Refactoring

Refactoring Jest Setup

- Create a new file `utils/testAppSetup.js` (or whatever you called the folder that you placed your test helpers in)
- Extract your `app` mocking/setup logic for Supertest into this file.
- Refactor your existing auth integration tests to utilize this new `testAppSetup.js` module for app setup. You will also use this for the new event routes integration tests.

This ensures consistent test setup across your integration tests and improves maintainability.